

Information Security

Prepared by Dr. Ahmed Omar

IT373
Information Security

Lecture 7



Application & Web Security

Lecture 7

IT373
Information Security

Prepared by
Dr. Ahmed Omar

Agenda

- ❑ Introduction to Web Application Security
- ❑ The Threat Landscape: OWASP Top 10
- ❑ Detecting and Mitigating Vulnerabilities
- ❑ Building Secure Applications: Best Practices

Web application security

- ❑ Web application security refers to the set of processes, technologies, and best practices designed to protect websites, web applications, and APIs from cyber threats, vulnerabilities, and unauthorized access.
- ❑ It focuses on defending applications that are accessed via browsers or APIs, ensuring that data remains *confidential*, services stay *available*, and *integrity* is maintained, even in the face of determined attacks

Web application security

Web Application Security is a specialized subset focusing on applications delivered through web browsers over HTTP/HTTPS protocols. This includes:

- ☐ Traditional web applications with server-side rendering
- ☐ Single Page Applications (SPAs) with client-side frameworks
- ☐ Web APIs and microservices
- ☐ Progressive Web Applications (PWAs)
- ☐ Hybrid mobile applications using web technologies

Why Web Application Security Matters

- ❑ **Widespread Exposure:** The world increasingly relies on web applications for banking, shopping, communication, and business operations. These applications are *globally accessible* and, *consequently, highly exposed to attackers*.
- ❑ **Prime Targets for Attackers:** Because web applications handle *sensitive data* and critical functions, attackers frequently exploit their vulnerabilities.

Why Web Application Security Matters

- ❑ **Business Impact:** Security breaches can lead to *data theft, service disruptions, financial losses*, damage to reputation, *loss of customer trust*, and even regulatory penalties if personal or financial data is compromised
- ❑ **Automated and Scalable Threats:** Attackers use automation to scan and exploit web applications at scale, searching for common vulnerabilities such as those in *code, configuration, business logic, or third-party services*

The Scale of the Problem

- *Financial Impact:*
 - ❑ *Average cost of a web application data breach: \$5.3 million (2025)*
 - ❑ *Web application attacks account for 94% of all cyber attacks*
 - ❑ *SQL injection alone affects 1 in 5 web applications globally*
 - ❑ *Cross-site scripting found in 84% of applications tested*

Challenges in Application Security

- ❑ **Rapid Release Cadence:** Continuous delivery/deployment increases chances of vulnerabilities slipping into production
- ❑ **Cloud & Hybrid Architectures:** Managing security across different environments is complex and requires proper controls
- ❑ **Human Factor:** Developer training is essential to reduce basic errors and insecure practices

Traditional vs. Modern Web Architecture

Modern web applications operate in a fundamentally different security context than traditional desktop applications:

1. Multi-tier Architecture Challenges:

- ❑ Presentation Layer: Browser-based clients with limited trust
- ❑ Application Layer: Server-side logic exposed through APIs
- ❑ Data Layer: Databases accessible through multiple pathways
- ❑ Network Layer: HTTP/HTTPS communication across untrusted networks

Common Web Application Security Threats

OWASP, which stands for the **O**pen **W**orldwide **A**pplication **S**ecurity **P**roject, is a global non-profit organization dedicated to improving software security. It operates as an open community, providing free and accessible resources to help organizations and individuals build and maintain secure software.

Founded in 2001, **OWASP's** mission is to make software security visible by providing unbiased, practical information and open-source tools. The organization is known for its community-led projects, global conferences, and extensive documentation that are available to anyone interested in application security.

OWASP

The Open Web Application Security Project (OWASP) maintains the industry standard for web security risks. The current OWASP Top 10 includes:

- A01- Broken Access Control**
- A02- Cryptographic Failures**
- A03- Injection Attacks**
- A04- Insecure Design**
- A05- Security Misconfiguration**
- A06- Vulnerable Components**
- A07- Authentication Failures**
- A08- Software Integrity Failures**
- A09- Security Logging Failures**
- A10- Server-Side Request Forgery**

OWASP

OWASP Top 10 Purpose:

- ☐ Awareness document for developers and web application security
- ☐ First step towards more secure coding
- ☐ Industry standard for web application security
- ☐ Foundation for security testing and code review

OWASP

A01 Broken Access Control

When users can act outside of their intended permissions, accessing unauthorized functionality or data.

How Attackers Do It:

They simply try to browse to URLs they shouldn't have access to (e.g., /admin or /user/123 by changing the ID to /user/456). They might manipulate parameters or exploit flaws in how the application checks permissions.

OWASP

A01 Broken Access Control

❑ *How Attackers Exploit:*

- URL manipulation: Changing `/user/profile/123` to `/user/profile/124`
- Parameter tampering: Modifying form fields or API parameters
- Directory traversal: Accessing files outside web root
- Privilege escalation: Regular user accessing admin functions

OWASP

A01 Broken Access Control

❑ Prevention:

- Implement proper authorization checks on every request
- Use role-based access control (RBAC)
- Apply principle of least privilege
- Regularly audit access permissions
- Use centralized access control mechanisms

OWASP

A02 Cryptographic Failures

This category focuses on failures related to cryptography that often lead to sensitive data exposure or system compromise. This includes issues like:

- Transmitting sensitive data in clear text (HTTP instead of HTTPS)
- Using weak or outdated cryptographic algorithms
- Using default, weak, or hardcoded cryptographic keys
- Not enforcing encryption based on proper security directives
- Improper certificate validation or weak cipher usage



OWASP

A02 Cryptographic Failures

Prevention

- Use strong, up-to-date encryption algorithms
- Implement HTTPS everywhere
- Proper key management and rotation
- Use established cryptographic libraries
- Regular security assessments

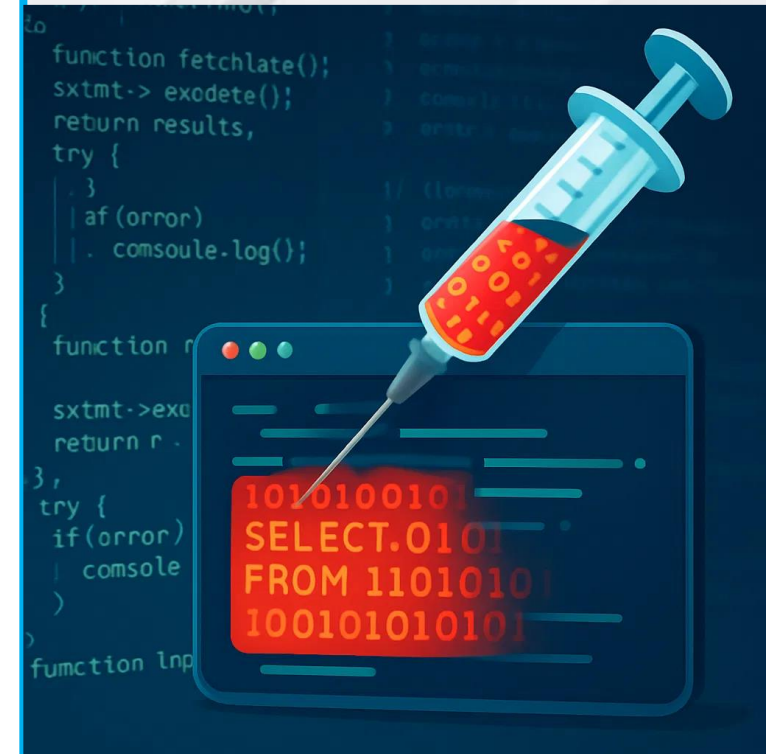
OWASP

A03: Injection

Injection flaws occur when untrusted data is sent to an interpreter as part of a command or query, tricking the interpreter into executing unintended commands or accessing unauthorized data.

Common Types:

- ❑ **SQL Injection:** Manipulating database queries
- ❑ **Cross-Site Scripting (XSS)**
- ❑ **Command Injection:** Executing system commands
- ❑ **LDAP Injection:** Manipulating directory service queries
- ❑ **XPath Injection:** Exploiting XML path language



OWASP

A03: Injection

Prevention

- ☐ Use parameterized queries/prepared statements
- ☐ Input validation and sanitization
- ☐ Principle of least privilege for database accounts
- ☐ Use stored procedures where appropriate
- ☐ Web Application Firewalls (WAF)

OWASP

A04 Insecure Design

Insecure design refers to flaws in the design and architecture of an application that cannot be fixed by perfect implementation. It represents a shift left in security thinking, focusing on risks and threats before coding begins.

Prevention:

- ☐ Establish secure development lifecycle with security professionals
- ☐ Use threat modeling for critical authentication, access control, business logic
- ☐ Integrate security language and controls into user stories
- ☐ Write unit and integration tests to validate security controls
- ☐ Segment system architecture by trust levels

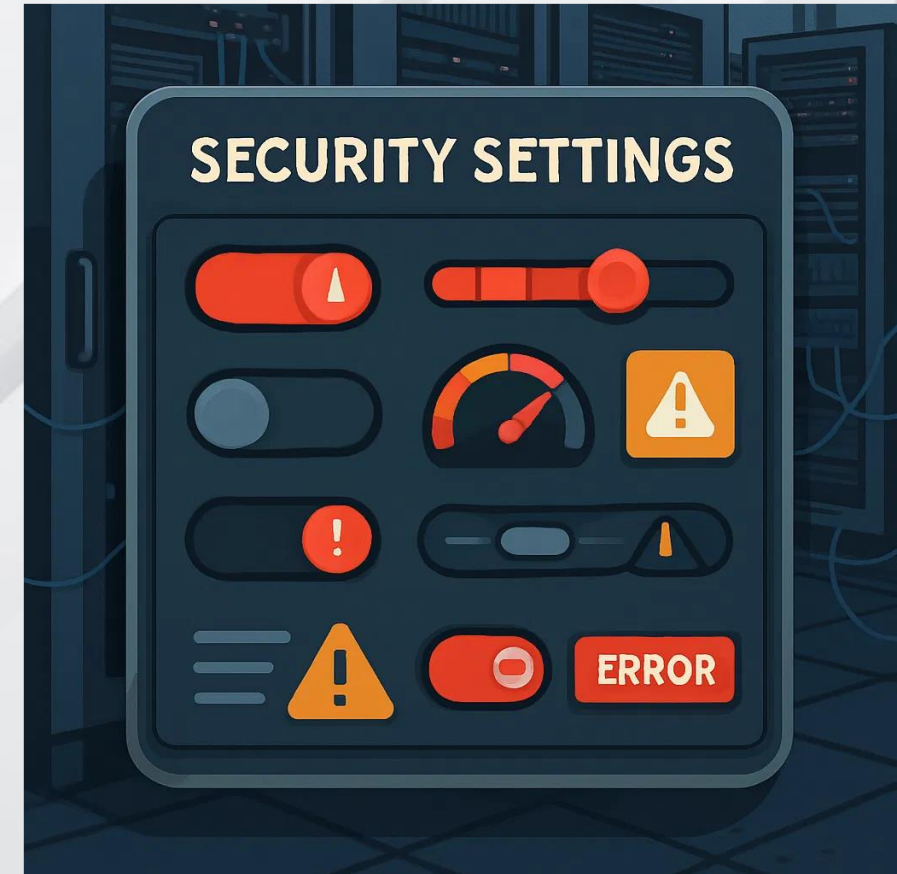
OWASP

A05 Security Misconfiguration

When security settings are defined, implemented, or maintained improperly.

Examples:

- ☐ Default accounts and passwords
- ☐ Unnecessary features enabled
- ☐ Missing security headers
- ☐ Verbose error messages revealing sensitive information

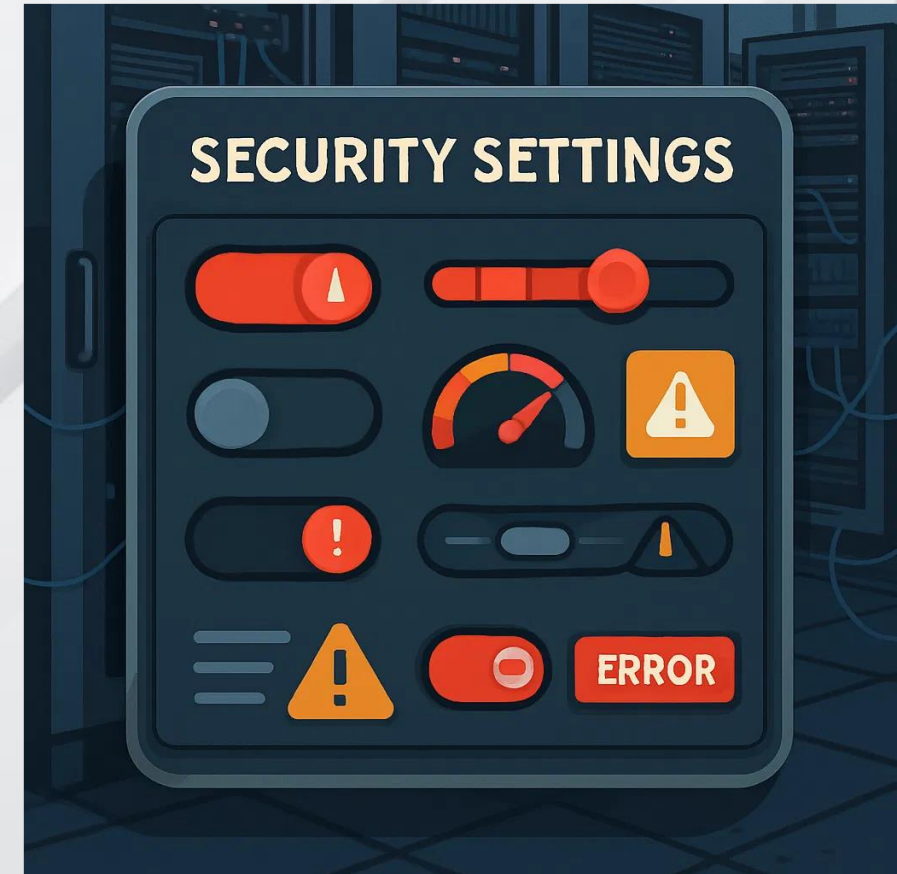


OWASP

A05 Security Misconfiguration

Prevention:

- ☐ Security hardening checklists
- ☐ Automated configuration scanning
- ☐ Regular security assessments
- ☐ Configuration management tools
- ☐ Principle of least functionality



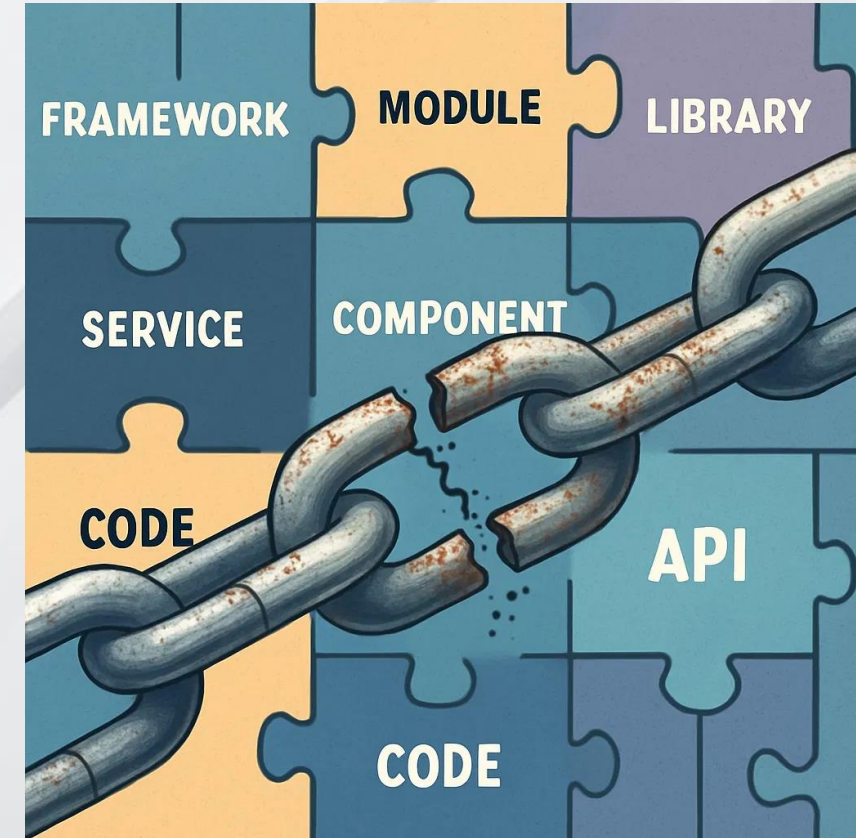
OWASP

A06 Vulnerable and Outdated Components

focuses on the risks associated with using components (libraries, frameworks, and other software modules) with known vulnerabilities.

Examples:

- ☐ Using components with known security vulnerabilities
- ☐ Lack of regular updates and patches for components
- ☐ Not testing compatibility of updated libraries
- ☐ Failure to scan for vulnerabilities regularly
- ☐ Not securing component configurations



OWASP

A06 Vulnerable and Outdated Components

Prevention:

- ☐ Continuously inventory versions of client-side and server-side components
- ☐ Monitor sources like CVE and NVD for vulnerabilities
- ☐ Use software composition analysis tools
- ☐ Obtain components from official sources over secure links

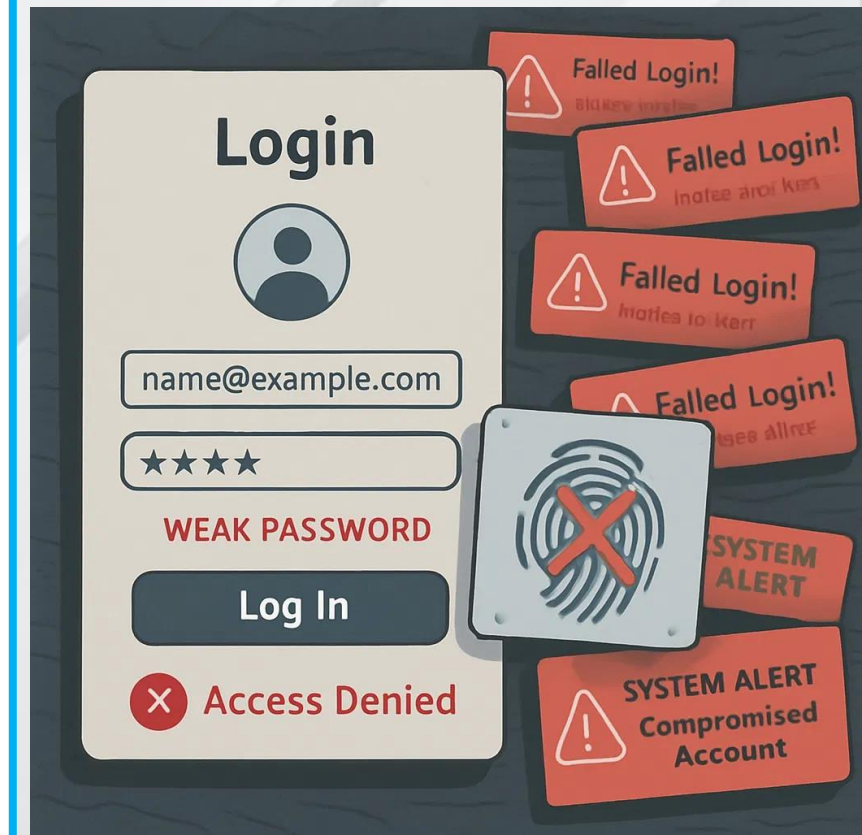
OWASP

A07 Identification and Authentication Failures

occur when functions related to user identity, authentication, or session management are implemented incorrectly.

Real-world Example

A web application allows unlimited login attempts without any account lockout mechanism, making it vulnerable to brute force attacks against user passwords.



OWASP

A07 Identification and Authentication Failures

Examples

- Permitting weak or default passwords
- Weak credential recovery and forgot-password processes
- Storing passwords in plain text or weakly hashed formats
- Missing or ineffective multi-factor authentication
- Exposing session IDs in URLs or not invalidating them properly

Prevention:

- Implement multi-factor authentication where possible
- Enforce strong password policies and checks
- Limit or delay failed login attempts
- Use secure session management with proper timeout and invalidation
- Test authentication mechanisms against common attack vectors

OWASP

A08 Software and Data Integrity Failures

Code and infrastructure that don't protect against integrity violations, including insecure deserialization.

Examples:

- ☐ Insecure deserialization of user data
- ☐ Unsigned software updates
- ☐ Lack of integrity checks on critical data

OWASP

A08 Software and Data Integrity Failures

Prevention

- ☐ Digital signatures for critical data
- ☐ Integrity checks and validation
- ☐ Secure deserialization practices
- ☐ Supply chain security measures
- ☐ Code signing for software distribution

OWASP

A09 Security Logging and Monitoring Failures

Insufficient logging, detection, monitoring, and active response to security events.

Prevention

- ☐ Comprehensive logging strategy
- ☐ Real-time monitoring and alerting
- ☐ Log integrity protection
- ☐ Security Information and Event Management (SIEM)
- ☐ Regular log review and analysis

OWASP

A10 Server-Side Request Forgery (SSRF)

When a web application fetches a remote resource without validating the user-supplied URL.

Example

- ☐ Accessing internal services through the application
- ☐ Reading local files via file:// schemes
- ☐ Scanning internal networks
- ☐ Bypassing firewalls and access controls

OWASP

A10 Server-Side Request Forgery (SSRF)

Prevention

- ☐ URL validation and allowlist
- ☐ Network-level controls
- ☐ Disable unnecessary URL schemes
- ☐ Regular security assessments

Application Security Testing Methods

To ensure applications are secure, various testing methodologies are employed throughout the development process

1. **Static Application Security Testing (SAST):** This method analyzes an application's source code, bytecode, or binaries for vulnerabilities without executing the application. SAST can be integrated early in the development lifecycle, helping developers identify and fix issues before deployment. It is considered a "white-box" testing approach.

Application Security Testing Methods

2. **Dynamic Application Security Testing (DAST):** This approach tests the application while it is running to find vulnerabilities in its live environment. DAST simulates real-world attacks and is effective at finding issues related to input validation and authentication. This is a "black-box" testing method.

Application Security Testing Methods

3. Interactive Application Security Testing (IAST): IAST combines elements of both SAST and DAST. It analyzes the code and the runtime environment simultaneously by integrating with the application as it runs, providing detailed insights.

Application Security Testing Methods

4. **Software Composition Analysis (SCA):** This type of testing focuses on identifying vulnerabilities in open-source components and third-party libraries used within an application.
5. **Runtime Application Self-Protection (RASP):** RASP operates within the application itself to monitor and defend against threats in real-time. It can identify and block attacks as they happen, offering a proactive layer of security.

OWASP

How to Perform OWASP Top 10 Penetration Testing

OWASP penetration testing focuses on the attack vectors identified in the Top 10. Security professionals simulate real-world attacks such as injection, broken authentication, and SSRF using automated tools (e.g., OWASP ZAP), manual review, and custom scripts.

Process Overview:

- Identify entry points and business logic.
- Execute tests for each Top 10 item.
- Document vulnerabilities with proof-of-concept.
- Provide actionable remediation guidelines.
- Present risk-impact analysis to stakeholders.

OWASP

How to Develop Secure Web Applications

- ✓ Train developers on the OWASP Top 10 risks and secure coding practices.
- ✓ Use secure coding guidelines and regular code reviews.
- ✓ Enforce principle of least privilege and robust access control.
- ✓ Integrate security tools (SAST, DAST) early in the build pipeline.
- ✓ Keep dependencies updated and monitor for vulnerabilities.
- ✓ Perform threat modeling and security testing throughout development life cycle

Summary

- ❑ Introduction to Web Application Security
- ❑ The Threat Landscape: OWASP Top 10
- ❑ Detecting and Mitigating Vulnerabilities
- ❑ Building Secure Applications: Best Practices



Thank You

